SVM-Based Speaker Verification System for Match-on-Card and Its Hardware Implementation

Woo-Yong Choi, Dosung Ahn, Sung Bum Pan, Kyo Il Chung,
Yongwha Chung, and Sang-Hwa Chung

Using biometrics to verify a person's identity has several advantages over the present practice of personal identification numbers (PINs) and passwords. To gain maximum security in a verification system using biometrics, the computation of the verification as well as the storing of the biometric pattern has to take place in a smart card. However, there is an open issue of integrating biometrics into a smart card because of its limited resources (processing power and memory space). In this paper, we propose a speaker verification algorithm using a support vector machine (SVM) with a very few features, and implemented it on a 32-bit smart card. The proposed algorithm can reduce the required memory space by a factor of more than 100 and can be executed in real-time. Also, we propose a hardware design for the algorithm on a field-programmable gate array (FPGA)-based platform. Based on the experimental results, our SVM solution can provide superior performance over typical speaker verification solutions. Furthermore, our FPGA-based solution can achieve a speed-up of 50 times over a software-based solution.

Keywords: Speaker verification, match-on-card.

I. Introduction

Traditionally, verified users have gained access to secure information systems, buildings, or equipment via multiple personal identification numbers (PINs), passwords, smart cards, and so on. However, these security methods have important weakness in that such items can be lost, stolen, or forgotten. In recent years, there has been an increasing trend of using biometrics, which refers to the personal biological or behavioral characteristics used for verification or identification [1], [2]. Biometrics relies on 'something that you are' and can inherently differentiate between a verified person and a fraudulent imposter. The problem of resolving the identity of a person can be categorized into two distinct types. Verification matches a person's claimed identity to his or her previously enrolled pattern (a 'one-to-one' comparison). However, identification identifies a person from the entire enrolled population by searching within a database for a match (a 'one-to-many' comparison).

In typical biometric verification systems, the biometric patterns are often stored in a central database. In a case where a high security level is needed, however, the database can be decentralized into millions of smart cards [3]-[5]. However, most of the current implementations of this solution have a common characteristic that the biometric verification process is solely accomplished out of the smart card. This system is called a Store-on-Card because the smart card is used only as a storage device to store a biometric pattern. That is, the biometric pattern stored in the smart card needs to be non-securely released into an external card reader to be compared with an input pattern.

To heighten the security level, the verification operation

Manuscript received Jan. 31, 2005; revised Feb. 22, 2006.

This study was supported in part by research funds from Chosun University, 2005.

Woo-Yong Choi (phone: + 82 42 860 1680, email: wychoi4@etri.re.kr), Dosung Ahn (email: dosung@etri.re.kr), and Kyo Il Chung (email: kyoil@etri.re.kr) are with Information Security Research Division, ETRI, Daejeon, Korea.

Sung Bum Pan (phone: + 82 62 230 6897, email: sbpan@chosun.ac.kr) is with the Department of Information, Control and Instrumentation Engineering, Chosun University, Gwangju, Korea.

Yongwha Chung (email: ychungy@korea.ac.kr) is with the Department of Computer Information, Korea University, Seochang, Chungnam, Korea.

Sang-Hwa Chung (email: shchung@pusan.ac.kr) is with the Department of Computer Engineering, Pusan National University, Busan, Korea.

needs to be performed by an in-card processor, not an external card reader [6]-[8]. This system is called a Match-on-Card because the verification operation is executed on the smart card. Note that standard PCs on which typical biometric verification systems have been executed have a 1 GHz CPU and 128 Mbytes of memory. On the contrary, a state-of-the-art smart card can at most employ a 50 MHz CPU, 64 Kbytes of ROM, 32 Kbytes of EEPROM, and 8 Kbytes of RAM. Therefore, typical biometric verification algorithms may not be executed on a smart card successfully.

Some examples of the biological characteristics are a subject's fingerprints, voice, face shape, iris, and vein distribution. Among these, the voice is one of the most promising biometrics because of its convenient use. There are many speaker verification algorithms such as dynamic time warping (DTW) [9], hidden Markov model (HMM) [10], Gaussian mixture model [11], and vector quantization [12]. These algorithms mainly focus on accuracy rather than execution time or memory requirement because they are used for resource-free environments such as a PC. Therefore, these algorithms may not be applied to a resource-constrained system such as a smart card.

A support vector machine (SVM), pioneered by Vapnik [13], is an example of a universal feed-forward network, and it has been widely used for pattern classification and non-linear regression in recent years. Wan [14] combined an SVM with a speaker verification task. However, this combination requires too much memory space to be executed in the smart card system.

Consider another situation where a customer claims his or her identity to a call center where a smart card cannot be used for verification. The call center should verify the customer's identity over the telephone, in which case speaker verification is the most convenient way. As there may be many verification requests simultaneously, a speaker verification system should be designed in-hardware to meet customer demands.

For this paper, we constructed an SVM-based speaker verification system with a very small amount of features, and implemented it in real-time on a 32-bit smart card. In general, speech features are extracted from each frame of utterance. Because typical speaker verification systems such as DTW and HMM use these features as they are, they cannot be executed in the smart card system. To meet the processing power and memory space specification of the smart card, we used the time average of all speech frames as a feature vector, which resulted in remarkable reductions in required memory space and execution time. In addition, we proposed a hardware design for an SVM-based speaker verification system on an FPGA-based platform for large-scale applications such as a call center. By carefully designing the required functions, we have implemented them on a Xilinx Virtex XCV600E. Using a clock rate of 50 MHz, the training and testing processes can be performed in 49.46 ms. The corresponding software solution could perform the same processes in 2,457 ms on a Pentium IV PC.

The organization of this paper is as follows. Section II explains the overview of our speaker verification system, and we briefly introduce an SVM for speaker verification in section III. Section IV explains the experimental results for a classifier decision, while we describe the implementation details of the smart card and the proposed memory-efficient SVM-based speaker verification algorithm in section V. Section VI discusses our hardware design of the SVM-based speaker verification. Finally, we make concluding remarks in section VII.

II. Overview of Speaker Verification

The speaker verification system shown in Fig. 1 has two phases: enrollment and verification. In the off-line enrollment phase, utterances from the reference speaker are preprocessed and the features are extracted, from which the speaker model is trained and stored. In the on-line verification phase, the similarity between the enrolled speaker model and the input pattern is examined.

There are two approaches in speaker verification: template-based and model-based approaches. DTW and HMM are the representatives of the former and the latter, respectively.

DTW performs a global time alignment procedure to compare speech patterns, which compensates for the different rates of speaking of two patterns. The dynamic path is chosen to minimize the accumulated distance along the piece-wise linear mapping path. The decision is made depending on this dissimilarity score and the predetermined threshold.

HMM is one of the well-known and widely used statistical methods of characterizing the spectral properties of the frame of a speech pattern. The underlying assumption of HMM is that the speech signal can be characterized as a parametric random process, and that the parameters of the stochastic process can be determined in a precise, well-defined manner. There is no known way to analytically solve for the model parameters that maximize the probability of the observation sequence in a closed form. We can, however, choose its likelihood function and locally maximize it using an iterative procedure such as the

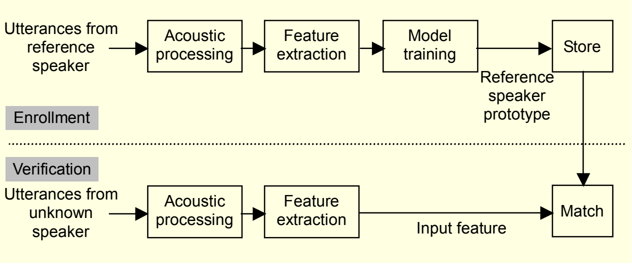

Fig. 1. Speaker verification system.

expectation-maximization (EM) algorithm [15].

In this paper, we used an SVM for a speaker verification algorithm. The basic idea of the SVM is to map the training data into a higher-dimensional feature space via a kernel, and to construct a separating hyperplane with maximum margin there, which yields a nonlinear decision boundary in the input space. A key idea that is central to the construction of the support vector learning algorithm is the inner-product kernel between the vectors drawn from the input space. Using the kernel function, we can compute the separating hyperplane without explicitly carrying out the mapping into the feature space. Depending on how this inner-product kernel is generated, we may construct different learning machines characterized by nonlinear decision surfaces of their own.

III. Speaker Verification Using an SVM

SVM is a binary classification method that finds the optimal linear decision surface based on the concept of structural risk minimization. The decision surface is a weighted combination of elements of a training set. These elements are called support vectors (SVs), which characterize the boundary between the two classes. For the purpose of explanation, we will briefly describe an SVM in the following. Details can be found in [16].

Consider a given set of examples,

$$(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N) \in \mathbf{R}^d \times \{\pm 1\}, \tag{1}$$

where x$_i$ is the input pattern for the *i*-th example and y_i is the corresponding desired response. For typical speaker verification systems, N is a very large number, and such systems are not feasible to be implemented on a smart card. In this paper, to implement speaker verification on a smart card, we used the time average of all speech frames as a feature vector, and the resultant size of N is equal to the number of training utterances. We first assume that the two classes are linearly separable for simplicity. Then, we can classify two classes by a hyperplane defined by

$$\mathbf{w}^T \mathbf{x} + b = 0 . \tag{2}$$

To train an SVM, we should find the hyperplane that has the maximum distance from the nearest data. That is, we should find a weight-vector w in (3) with property (4).

$$\min \ \frac{1}{2} \mathbf{w}^T \mathbf{w} \tag{3}$$

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, N \tag{4}$$

By using a Lagrange formula we can express (3) and (4) as

$$\max \ Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{5}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ \text{ and } \ \alpha_i \geq 0 \ \text{ for } \ i = 1, \cdots, N \tag{6}$$

If α_o makes $Q(\alpha)$ maximum, then the optimal solution of the weight vector $\mathbf{w}_o$ is

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} y_i \mathbf{x}_i , \tag{7}$$

and we get the optimal bias b_o using following equation

$$b_o = 1 - \mathbf{w}_o^T \mathbf{x}^{(s)} \ \text{ for } \ y^{(s)} = 1 , \tag{8}$$

where $\mathbf{x}^{(s)}$ is the input pattern whose Lagrange multiplier is non-zero.

In the preceding explanation, we have focused on linearly separable patterns. Now, we consider the case of linearly non-separable patterns. The basic idea of a non-linear SVM is to perform non-linear mapping of an input vector into high dimensional dot product space *F*, which is called a feature space.

Let $\varphi(\mathbf{x})$ be a non-linear mapping from input space to feature space, then the optimal hyperplane is defined by

$$\sum_{i=1}^{N} \alpha_i y_i \varphi^T(\mathbf{x}) \varphi(\mathbf{x}_i) = 0 . \tag{9}$$

In general, however, the dimension of the feature space is very large. Thus, we have the technical problem of computing matrix multiplications in the high dimensional space. A Kernel method can solve this problem. The inner-product kernel is defined by

$$k(\mathbf{x}, \mathbf{x}_i) = \varphi^T(\mathbf{x}) \varphi(\mathbf{x}_i) . \tag{10}$$

Substituting (10) in (9) and using a Lagrange multiplier, the objective function shown in (5) and (6) is now defined by

$$\max \ Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) , \tag{11}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ \text{ and } \ \alpha_i \geq 0 \ \text{ for } \ i = 1, \cdots, N . \tag{12}$$

Differentiating $Q(\alpha)$ with respect to the Lagrange multipliers yields the following set of simultaneous equations:

$$\mathbf{K}'\alpha' = \mathbf{y}', \tag{13}$$

and we can get α_i's using a Gauss-Jordan algorithm:

$$\mathbf{K}' = \begin{bmatrix} 0 & y_1 & y_2 & \cdots & y_N \\ 1 & k_{11} & k_{12} & \cdots & k_{1N} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & k_{N1} & k_{N2} & \cdots & k_{NN} \end{bmatrix}, \; \alpha' = \begin{bmatrix} \lambda \\ \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \; \mathbf{y}' = \begin{bmatrix} 0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix}, \tag{14}$$

where $k_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$ and λ is a Lagrange multiplier of the constraint (12).

For speaker verification, we use the following radial-basis function (RBF) kernel (selection of this kernel will be discussed in section IV):

$$k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma^2} \| \mathbf{x} - \mathbf{x}_i \|^2 \right). \tag{15}$$

IV. Experimental Results for Classifier Decision

To choose the implementation details of the SVM solution, we first conducted experiments with various kernels on a Pentium IV PC [17]. Also, to evaluate the effectiveness of the SVM solution, we compared the recognition accuracy, model size, and execution time of the SVM solution with those of typical speaker verification solutions such as DTW and HMM.

We used a Korean database from Pusan National University [18] for the experiments. The database consists of four-digit strings, isolated words, and short sentences recorded in an office environment. After the end-point detection was finished, the average time durations of the four-digit strings, isolated words, and short sentences were 1.5, 1.48, and 2.06 seconds, respectively. The speech was coded into 20 ms frames, with a frame advance of 10 ms. Each frame was represented by twelve Mel-frequency Cepstral coefficients and their deltas. The sampling rate of the speech was 16 KHz, and the quantizing rate was 16 bits.

For our experiment, we used a continuous density HMM. Each utterance was modeled by a single left-to-right HMM, and the number of states varied with the length of utterance. The observation distribution for each state was modeled by a multivariate Gaussian mixture distribution with six mixtures, and each of the mixture components had a common diagonal covariance matrix.

The data used for training models consist of six repetitions for each utterance from 27 speakers (seventeen males and ten females). The test portion of the database consists of six repetitions of the same words used for training models from each speaker. All the experiments were conducted in a text-dependent mode.

The speaker verification performance of the SVM with various kernels is shown in Table 1. The RBF kernel shows the lowest total-error rate (TER), which is defined in equation (16), whereas the polynomial kernel with a degree of 3 shows the highest TER.

$$\text{TER} = \text{FAR} + \text{FRR} \tag{16}$$

In the case of a polynomial kernel, the higher the degree of polynomial, the lower the performance we could achieve. The reason for this performance degradation was that the system could not make models for many speakers since the training data were insufficient for training high-degree polynomial models. The performance using the RBF kernel also varied with the parameter values. We achieved the lowest TER when we used the RBF kernel with the standard deviation of 8. In the following, we used this configuration for further evaluation.

We compared the SVM with typical speaker verification algorithms. Table 2 shows the TERs of the SVM, DTW, and HMM. These results indicate that the SVM can outperform DTW and HMM. In particular, compared to HMM, the most widely-used algorithm in speaker verification, the SVM solution can reduce the error rate by a factor of two.

We also compared them in terms of execution times and

Table 1. Total error rates of SVMs with various kernels.

Kernel	TER (%)
Linear	2.91
Polynomial (degree = 1)	2.97
Polynomial (degree = 2)	20.8
Polynomial (degree = 3)	78.0
RBF (standard deviation = 2)	3.59
RBF (standard deviation = 4)	2.31
RBF (standard deviation = 6)	1.79
RBF (standard deviation = 8)	1.76
RBF (standard deviation = 10)	1.79
RBF (standard deviation = 20)	2.50
RBF (standard deviation = 30)	3.42

Table 2. Total error rates of SVM, DTW and HMM.

Algorithm	TER (%)
SVM (RBF kernel)	1.76
DTW	5.14
HMM	4.70

model sizes. We measured the execution times on a Pentium IV (1.3 GHz) PC running Windows 2000. Table 3 shows that the average training time of the SVM is longer than that of DTW, and is shorter than that of HMM. The average testing time of the SVM, however, is much shorter than that of both DTW and HMM.

The model sizes of the SVM, DTW, and HMM depend on the number of support vectors, total frames, and HMM states, respectively. Table 4 shows the average model sizes of the three algorithms. While the model sizes of DTW and HMM were 16.8 Kbytes and 4.6 Kbytes, respectively, the SVM used only 1.6 Kbytes to store the model. Therefore, the SVM has a great advantage in either large applications where millions of people are enrolled or smart card applications where only a few Kbytes of RAM is available. By using this memory-efficient SVM algorithm, we have successfully realized a Match-on-Card system for speaker verification.

Table 3. Execution times of SVM, DTW, and HMM.

Algorithm	Execution time (ms)	
	Training	Test
SVM (RBF kernel)	2,455	2
DTW	278	28
HMM	12,515	17

Table 4. Model sizes of SVM, DTW, and HMM.

Algorithm	Model size (kB)
SBM (RBF kernel)	1.6
DTW	16.8
HMM	4.6

V. Implementation on a Smart Card

Figure 2 shows the smart card system we are developing, and its characteristics are summarized as follows.

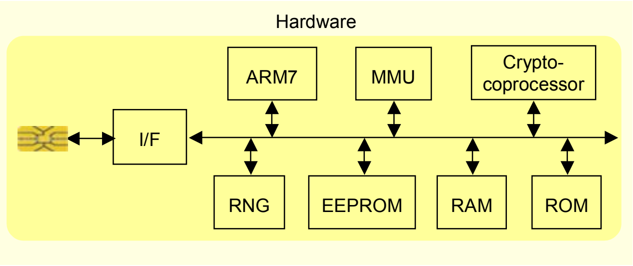

Fig. 2. Targeted smart card system.

1. Smart Card

The in-card processor to manipulate and interpret data is a 32-bit ARM7TDMI. The memory in the smart card consists of three different types. ROM is used for the card operating system (COS) and is usually embedded during manufacturing.

RAM is used by the COS as a temporary storage area. The user available data segments are allocated in EEPROM. Table 5 shows the system specification of the smart card that we are developing for a match-on-card [4]. The size of each memory type is 64 Kbytes, 8 Kbytes, and 32 Kbytes for ROM, RAM, and EEPROM, respectively. The first two types of memory are not available for user access. Several levels of access security are supported in the EEPROM. The methods of assigning access security can be controlled through use of a PIN, biometric information, or by using cryptography. The smart card also includes a crypto-coprocessor and random number generator (RNG) to perform cryptographic algorithms in real-time. Finally, for the contact interface, the external interface module is included.

Note that, because of the constrained size of the smart card chip, we select an ATM7TDMI. However, the maximum performance of the in-card processor is 60 million instructions per second, and its maximum clock rate is 66 MHz. This processing power is very limited compared with the typical PCs having 1 GHz. Thus, a careful performance analysis is required to integrate the biometrics into the resource-constrained smart card system. Figure 3 shows a photograph of the evaluation system.

Table 5. System specification of the smart card.

CPU	32-bit RISC processor (ARM7TDMI)
ROM	64 kB
RAM	8 kB
EEPROM	32 kB

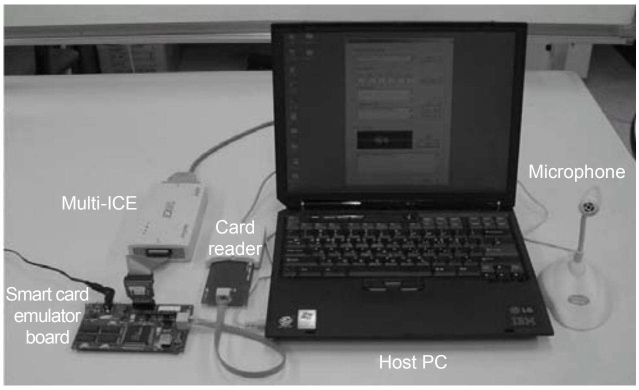

Fig. 3. Photograph of the evaluation system.

2. Speaker Verification on Smart Card

Figure 4 shows a Match-on-Card for speaker verification. Note that the matching step to compute the similarity between the speaker model and the input feature is executed on the Match-on-Card, whereas the acoustic processing and feature extraction steps are executed on the card reader. In the off-line enrollment phase, acoustic processing, feature extraction, and model training steps are executed in the card reader, and the resulting model is stored in the smart card. In the on-line verification phase, the features extracted from an utterance from an unknown user are transferred to the smart card, where the similarity between the enrolled speaker model and input feature is then examined.

In the feature extraction step, the speech signal is blocked into frames, and each individual frame is windowed so as to minimize the signal discontinuities at the beginning and end of each frame. Speech features are then extracted from each frame of the windowed signal. In general, there are hundreds of frames in an utterance, so thousands of features are needed to represent every single utterance. This does not meet the memory space specification of the smart card. In this paper, we used the time average of all speech frames as a feature vector, so we can represent each utterance by a single 24-dimensional vector. This is small enough to be processed in the smart card. Furthermore, there are remarkable reductions in execution times and memory requirements for both enrollment and verification.

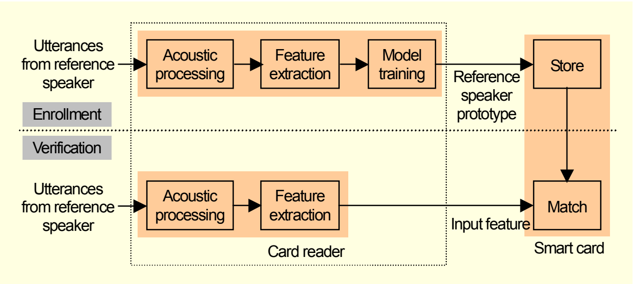

Fig. 4. Match-on-Card for speaker verification.

VI. A Hardware Implementation for Speaker Verification

As shown in Fig. 5, the architecture of a speaker verification system using an SVM consists of the following five logical modules:

- Train vector matrix multiplication (VMM) controller
- Kernel function controller
- Gauss Jordan controller
- Support vector table controller
- Test VMM controller

A flowchart of a speaker verification algorithm using an SVM is shown in Fig. 6. A speaker verification system using an SVM should handle two processes: *training* and *testing*. The training process is the process that generates SVs. On the contrary, the testing process is the process that verifies the test vector using the SVs. The training process computes a **K′** matrix first, then computes the Lagrange multiplier, and finally generates an SV-table. The train VMM and kernel function controllers take charge of computing the **K′** matrix. That is, the train VMM controller transfers addresses of the stored training vectors in SRAM to the kernel function controller. After reading the training vectors using the addresses given from the VMM controller, the kernel function controller executes the RBF kernel function. The train VMM controller computes the **K′** matrix with the results returned from the kernel function controller.

The Gauss Jordan controller takes charge of computing the Lagrange multiplier using the **K′** matrix. The Gauss Jordan controller is composed of a *Big Finder*, *Swap*, *Big Row*, and *Matrix Calculator*, and computes the Lagrange multiplier by using the Gauss Jordan algorithm. The support vector table controller generates the SV-table from the Lagrange multiplier. Finally, the test VMM controller computes the similarity between the enrolled SVs and test vector.

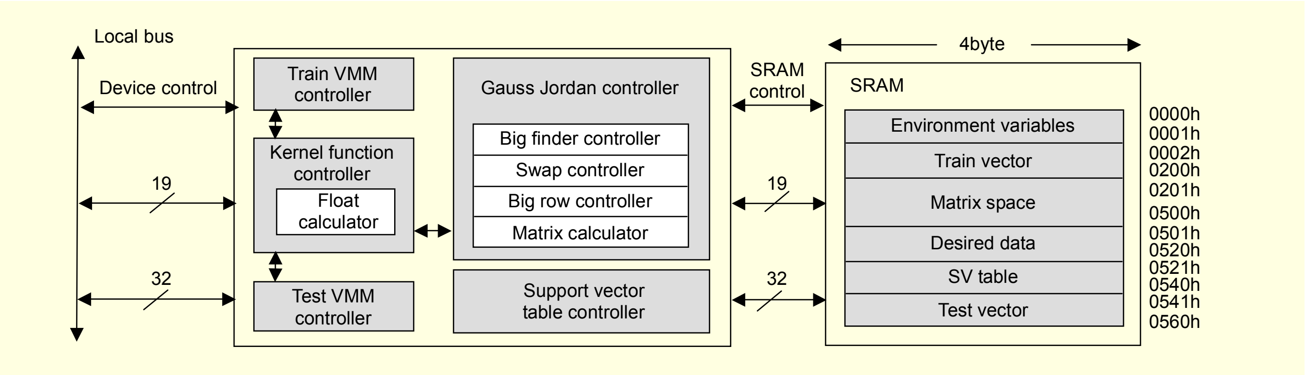

Fig. 5. The architecture of speaker verification system.

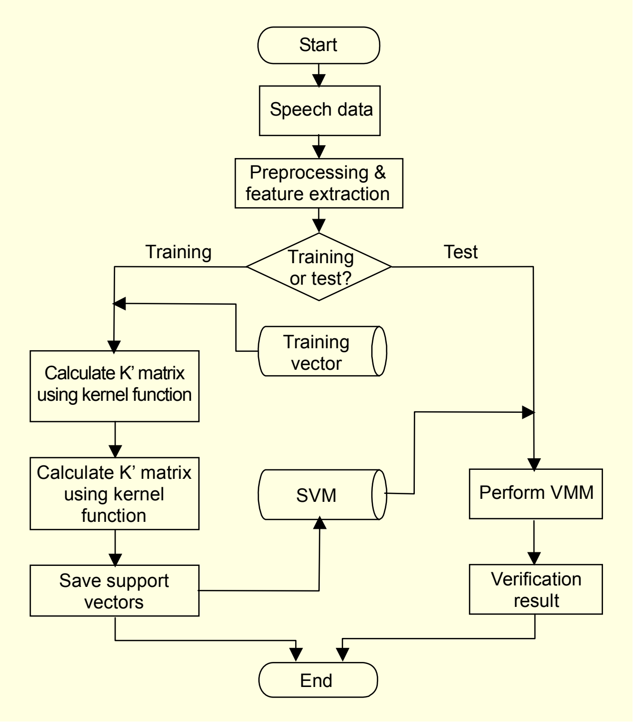
Start
Speech data
Preprocessing & feature extraction
Training
Training or test?
Test
Training vector
Calculate K' matrix using kernel function
Calculate K' matrix using kernel function
SVM
Perform VMM
Save support vectors
Verification result
End

Fig. 6. Flowchart of speaker verification algorithm using SVM.

1. Training Process

The training process executed by the train VMM controller is a process that obtains the matrix of **K**′ using the training vector described by (14) in section III. Since the dimension of the training vector is variable, it is impossible for the system to operate with a fixed configuration. Therefore, after storing some environment variables in the SRAM, our system should set up the configurations. That is, the total number of vectors and their dimensions are described by those environment variables without considering the dimension of the training vector.

RBF Kernel Function. As described in section III, the kernel function used in this hardware design is an RBF kernel function that includes an exponential equation. However, it is time-consuming to compute the exponential equation directly in-hardware. Thus, the system includes a table that turns the result of the RBF kernel function to a specially fixed value corresponding to the result of the exponential equation. Then, the kernel function is used to compute a new **K**′ matrix in the training process and verify the test vector in the testing process. The main operation of the kernel function is an inner product of the vector. The inner product of the *d*-dimension vector is composed of four arithmetical operations with real numbers, which is executed in a float calculator.

Lagrange Multiplier. As described in section III, the Lagrange multiplier α_i shown in (12) to (15) has to be computed to obtain the SVs. The Gauss Jordan controller computes the Lagrange multiplier after generating the **K**′ matrix. The **K**′ matrix was derived from the result of the RBF kernel function using training vectors. The operational steps performed by the Gauss Jordan controller are as follows:

First, it can find the maximum value in the **K**′ matrix using *Big Finder*. It then exchanges the maximum row to the row corresponding to the maximum column using *Swap*. Note that the maximum row means the row including the maximum value, and the maximum column means the column including the maximum value. Then, *Big Row* computes the maximum row, and the matrix calculator computes the other rows. Finally, it can obtain the Lagrange multipliers after repeating *N* times.

SV-table. The Gauss Jordan module writes α_i as well as the **K**′ matrix in the SRAM. Note that α_i is a Lagrange multiplier of each training vector. Then, the support vector table controller reads the Lagrange multipliers from the SRAM. If the 32nd-bit value of the Lagrange multiplier is 0, then the corresponding training vector is named an SV. The SV-table is generated with the address and Lagrange multiplier of the SVs. The support vector table controller stores the start address of the SV-table and the number of SVs in the SRAM.

2. Testing Process

The testing process runs in the test VMM controller. After reading the SV-table, the test VMM controller can obtain the number of SVs and the addresses of the SVs corresponding to the claimed speaker's ID. Also, the test VMM controller

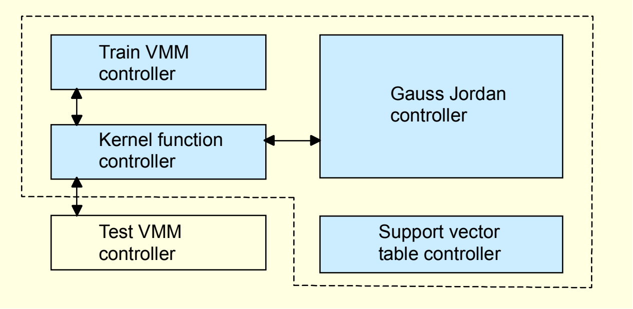

Fig. 7. Block diagram of the training process.

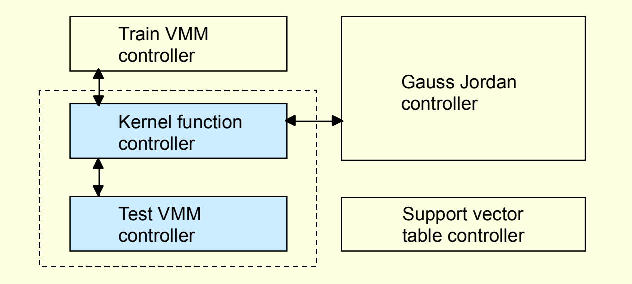

Fig. 8. Block diagram of the testing process.

computes the similarity between the enrolled SVs and the test vector using the kernel function for verification.

Based on the evaluation results with a software-based SVM solution, we have implemented a hardware-based SVM system with RBF kernel. As shown in Fig. 9, our system is composed of ARM9, Xilinx FPGA, SRAM, and the PCI I/F to communicate with a host.

We chose the dimension of vector data and the number of training vectors as 24 and 31, respectively. Through further evaluation, we confirmed that our hardware system formed an SV-table, found the SVs, and verified the test vector correctly.

Also, we adopted Xilinx Virtex XCV600E as our FPGA, and the usage rate of the slice was 90 percent. Thus, the number of gates of our design was 508,845 as shown in Table 6. When the system operated in 50 MHz, the training process time was 48.8 ms and the testing process time was 0.66 ms. Our FPGA-based solution can achieve a speed-up of close to 50 times compared to a software-based SVM solution. To the best of our knowledge, our solution is the first hardware-based SVM solution for speaker verification, and can be used for large-scale applications such as customer verification in call centers.

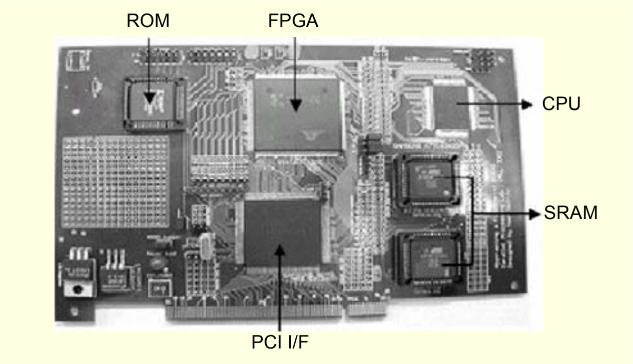

Fig. 9. Speaker verification hardware system using SVM.

Table 6. Required resources in Xilinx Virtex XCV600E.

Number of slices	6,526 out of 6,912 (94%)
Number of slice flip flops	10,183 out of 13,824 (74%)
Total number 4 input LUTs	9,974 out of 13,824 (72%)
Total equivalent gate count for design	**508,845**

VII. Concluding Remarks

The smart card is a model of a very secure device, and biometrics is a promising technology for verification. These two can be combined for many applications to enhance both security and convenience. However, typical biometric verification algorithms that have been executed on standard PCs may not be executed in real-time in a resource-constrained environment.

In this paper, we have presented a memory-efficient SVM-based speaker verification algorithm that can be executed in real-time on a smart card. The conventional version of an SVM requires about 183 Kbytes to store a speaker model, and the execution times for training and testing are 353 seconds and 58.7 milliseconds, respectively. It is not feasible, however, to implement the SVM in the smart card. To meet the processing power and memory space specification of the smart card, we used the time average of all speech frames as a feature vector, which resulted in remarkable reductions in required memory space and execution time. Consequently, we have successfully ported our speaker verification algorithm to the ARM7. Also, we have shown an FPGA-based design for speaker verification using SVM. To choose the implementation details of the SVM solution, we first conducted experiments with various kernels on a Pentium IV PC. Also, we compared the accuracy, model size, and execution time of the SVM solution with those from typical speaker verification solutions. Based on the experimental results, our FPGA-based solution using a fixed-point operation can achieve a speed-up of 50 times over a software-based solution using a floating-point operation, and can be used in large-scale applications.

Since an SVM is a very general classification technique, our hardware design can be applied to other biometrics such as a face, fingerprint, and iris. Also, parallel processing techniques can be employed for further improvement in throughput.

References

[1] A. Jain, R. Bole, and S. Panakanti, *Biometrics: Personal Identification in Networked Society*, Kluwer Academic Publishers, 1999.

[2] Y. Seto, “Personal Authentication Technology Using Biometrics,” *SICE*, vol. 37, no. 6, 1998, pp. 395-401.

[3] H. Dreifus and T. Monk, *Smart Cards*, John Wiley & Sons, 1997.

[4] G. Hachez, F. Koeune, and J. Quisquater, “Biometrics, Access control, Smart cards: A Not So Simple Combination,” *Proc. 4th Working Conf. on Smart Card Research and Advanced Applications*, 2000, pp. 273-288.

[5] B. Struif, “Use of Biometrics for User Verification in Electronic Signature Smartcards,” *Proc. E-smart* 2001, LNCS 2140, 2001, pp. 220-227.

[6] R. Sanchez-Reillo, “Smart Card Information and Operations Using Biometrics,” *IEEE AEES Mag*., 2001, pp. 3-6.

[7] R. Sanchez-Reillo, J. Liu-Jimenez, L. Entrena, “Architectures for Biometric Match-on-Token Solutions,” *Proc. BioAW* 2004,

LNCS 3087, 2004, pp. 195-204.

[8] S.B. Pan, Y.H. Gil, D. Moon, Y. Chung, and C.H. Park, "A Memory-Efficient Fingerprint Verification Algorithm Using a Multi-Resolution Accumulator Array," *ETRI Journal*, vol. 25, no. 3, June 2003, pp. 179-186.

[9] M. Pandit and J. Kittler, "Feature Selection for a DTW-Based Speaker Verification System," *Proc. ICASSP*, vol. 2, 1998, pp. 769-772.

[10] L. Raniber and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[11] C.M. del Alamo, F.J. Caminero Gil, C. dela Torre Munilla, and L. Hernandez Gomez, "Discriminative Training of GMM for Speaker Identification," *Proc. ICASSP*, vol. 1, 1996, pp. 89-92.

[12] J. He, L. Liu, and G. Palm, "A New Codebook Training Algorithm for VQ-Based Speaker Recognition," *Proc. ICASSP*, vol. 2, 1997, pp. 1091-1094.

[13] B. Scholkopf, K.-K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers," *IEEE Trans. on Signal Processing*, vol. 45, no. 11, 1997, pp. 2758-2765.

[14] V. Wan and W.M. Campbell, "Support Vector Machines for Speaker Verification and Identification," *Proc. IEEE Workshop on Neural Networks for Signal Processing*, vol. 2, Dec. 2000, pp. 775-784.

[15] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, 1977, pp. 1-38.

[16] B. Scholkopf, C. Burges, and A. Smola, *Advances in Kernel Methods*, The MIT Press, 1999.

[17] W. Choi, K. Lee, and Y. Chung, "Support Vector Machines for Robust Speaker Verification," *Proc. AICSST*, 2002, pp. 262-267.

[18] http://voice.ee.pusan.ac.kr

Woo-Yong Choi received the BS degree in statistics and the MS degree in electronics engineering from Pusan National University, Korea in 1998 and 2000. From 2000 to 2001, he worked for L&H Korea. Since joining ETRI in 2001, he has been working as a senior member of engineering staff at the Biometrics Technology Research Team. His research interests are in biometrics, speech recognition, and pattern recognition.

Dosung Ahn received the BS degree in automation engineering in 1992, the MS degree in mechanical engineering in 1994, and PhD degree in automation engineering in 2001, all from Inha University, Korea. He joined ETRI in 2001, and he is currently a senior member of engineering staff at Biometric Technology Research Team. His research interests are in biometrics, pattern recognition, and security.

Sung Bum Pan received the BS, MS, and PhD degrees in electronics engineering from Sogang University, Korea, in 1991, 1995, and 1999. He was a Team Leader at Biometric Technology Research Team of ETRI from 1999 to 2005. He is now a Full-time Instructor at Chosun University. His current research interests are in biometrics, security, and VLSI architectures for real-time image processing.

Kyo Il Chung received the BS, MS, and PhD degrees in electronic engineering from Hanyang University in 1981, 1983, and 1997. He joined ETRI in 1982 and has been involved with COMSEC systems. Currently, he is a principal member of engineering staff and his role is Director of Information Security infrastructure Research Group. His research interests are in IC cards, RFID, biometrics, and information warfare.

Yongwha Chung received the BS and MS degrees from Hanyang University, Korea, in 1984 and 1986. He received the PhD degree from the University of Southern California, USA in 1997. He worked for ETRI from 1986 to 2003 as a Team Leader. Currently, he is an Associate Professor in the Department of Computer Information, Korea University. His research interests include biometrics, security, and performance optimization.

Sang-Hwa Chung received the BS degree in electrical engineering from Seoul National University in 1985, the MS degree in computer engineering from Iowa State University in 1988, and the PhD degree in computer engineering from the University of Southern California in 1993. He was an Assistant Professor in the Electrical and Computer Engineering Department at the University of Central Florida from 1993 to 1994. He is currently a Professor in the Computer Engineering Department at Pusan National University, Korea. His research interests are in the areas of computer architecture and high-performance computer networking.